2021-03

IDENTIFICATION AND CLASSIFICATION OF SIGNALS USING GENERATIVE ADVERSARIAL NETWORKS

Ellison, Bart D.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IDENTIFICATION AND CLASSIFICATION OF SIGNALS USING GENERATIVE ADVERSARIAL NETWORKS

by

Bart D. Ellison

March 2021

Thesis Advisor: Frank E. Kragh
Co-Advisor: James W. Scrofani

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE IDENTIFICATION AND CLASSIFICATION OF SIGNALS USING GENERATIVE ADVERSARIAL NETWORKS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Bart D. Ellison	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Research has shown that machine learning holds promise as a technique to improve the identification and classification of signals of interest. This study proposes the use of machine learning, specifically generative adversarial networks, to classify received signals based on their down-converted, but not demodulated, in-phase and quadrature signals and evaluate their probability of being of interest. The approach used a generative adversarial network to train a classifier convolutional neural network to determine the likelihood that a received signal is of interest. We tested the ability of a semi-supervised generative adversarial network to classify signals of interest by modulation scheme. We then tested the ability of the semi-supervised generative adversarial network to identify unique signals of interest within a dataset of a single modulation scheme. We evaluated the performance of the network on accuracy, training time, and the amount of data needed to train the network. The results proved that a semi-supervised generative adversarial network could classify a signal by modulation scheme and identify signals within a single modulation scheme.

14. SUBJECT TERMS generative adversarial network, neural network, deep learning	**15. NUMBER OF PAGES** 77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

IDENTIFICATION AND CLASSIFICATION OF SIGNALS USING GENERATIVE ADVERSARIAL NETWORKS

Bart D. Ellison
Lieutenant Commander, United States Navy
BSEE, Louisiana Tech University, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2021

Approved by: Frank E. Kragh
Advisor

James W. Scrofani
Co-Advisor

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Research has shown that machine learning holds promise as a technique to improve the identification and classification of signals of interest. This study proposes the use of machine learning, specifically generative adversarial networks, to classify received signals based on their down-converted, but not demodulated, in-phase and quadrature signals and evaluate their probability of being of interest. The approach used a generative adversarial network to train a classifier convolutional neural network to determine the likelihood that a received signal is of interest. We tested the ability of a semi-supervised generative adversarial network to classify signals of interest by modulation scheme. We then tested the ability of the semi-supervised generative adversarial network to identify unique signals of interest within a dataset of a single modulation scheme. We evaluated the performance of the network on accuracy, training time, and the amount of data needed to train the network. The results proved that a semi-supervised generative adversarial network could classify a signal by modulation scheme and identify signals within a single modulation scheme.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AC-GAN	Auxiliary Classifier Generative Adversarial Network
CNN	Convolutional Neural Network
COMINT	Communication Intelligence
dB	Decibel
I/Q	In-phase and Quadrature
GAN	Generative Adversarial Network
LSTM RNN	Long Short Term Memory Recurrent Neural Network
NN	Neural Network
QPSK	Quadrature Phase Shift Keying
RadioML Dataset	Radio Machine Learning Dataset
ReLU	Rectified Linear Unit
RN	Residual Network
RNN	Recurrent Neural Network
SGAN	Semi-supervised Generative Adversarial Network
SNR	Signal-to-Noise Ratio
SOI	Signal of Interest

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Communication Intelligence (COMINT), often involving the collection of wireless communication signals, is an invaluable resource to the military and the intelligence community. The growth of wireless technologies and usage have resulted in a radio frequency spectrum with a high density of signals. Identifying Signals of Interest (SOIs) in such an environment can be challenging and labor-intensive. Advances in the application of machine learning may offer a more efficient means to identify and classify SOIs. Often, identifying SOIs is time-consuming, requiring significant human interaction in the process. Advances in the application of machine learning may help to reduce the burden of identifying SOIs.

In recent years, researchers have begun using machine learning to improve radio efficiency through spectrum management. Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN) have been explored to classify nearby signals by modulation type within Cognitive Radio Networks [1]–[12]. These technologies show potential as a way to classify and identify SOIs.

Within the large amount of data collected, the SOIs typically comprise a tiny portion. Neural networks (NNs) require large, labeled datasets to train, which we may not have of an SOI. However, it requires significantly less data to train a GAN, making them a viable option when classifying SOIs [11]. This thesis will explore using a GAN to identify SOIs with limited training data and limited preprocessing of the input signals.

A. PROBLEM STATEMENT

Collected communications signals can be a valuable source of intelligence. However, not all the signals collected are of importance to the intelligence community. A large amount of raw communication signal data requires significant time and effort to process and identify the vital SOIs. To reduce this time and resource-intensive process, we propose using a semi-supervised GAN (SGAN) to reduce the time and human resources needed to identify and classify SOIs.

B. RESEARCH QUESTIONS

We pose the following questions to prove the viability of using the SGAN to identify SOIs with a small amount of training data and minimum preprocessing of the data signals. The first set of questions address the classification of an SOI defined by its modulation type. The second set of questions addresses the SGAN ability to identify an SOI within a single modulation scheme.

1. Can an SGAN classify signals by modulation type when given training data in in-phase and quadrature form?
    - What is the required amount of training data required to enable accurate classification?
    - What is the accuracy at different signal-to-noise ratios (SNRs)?
    - How long does it take to train the GAN?
2. Given a dataset of one type of signal modulation, can an SGAN classify the SOI?
    - What is the required amount of training data required to enable accurate classification?
    - What snippet length is required to enable accurate classification?
    - What is the accuracy at different signal-to-noise ratios (SNRs)?
    - How long does it take to train the GAN?

C. MOTIVATION

As wireless communication continues to grow at an exponential pace, the ability to quickly and efficiently turn collected data into actionable intelligence will also grow. The overwhelming amount of collected signals continue to need to be classified and sorted to identify SOIs. This thesis goal is to develop a GAN that could ease this burden and lead to partial automation.

D. THESIS ORGANIZATION

The rest of this thesis is structured as follows. Chapter II introduces neural networks, including convolutional NNs (CNNs) and GAN theory. Chapter III covers the previous uses of neural networks in signal classification and the current use of GAN technology. Chapter IV describes the two different data sets used in the research and the software tools used to create the GAN. Chapter V discusses the GAN design and implementation and includes the GAN architecture, performance measures, and experiment setup. Chapter VI summarizes the results and recommends future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. THEORY

A. NEURAL NETWORKS

NNs are a subset of machine learning loosely modeled after an organic brain. The purpose of a NN is to learn from data examples without being explicitly programmed [13].

1. Artificial Neurons

The basic building block of the neural network is the artificial neuron or perceptron. The artificial neuron consists of input nodes multiplied by weights [14]. The weights on the input nodes are a vital part of the artificial neuron as they are variable and determine how much each input affects the artificial neuron's output. An activation function is then applied to the weighted sum of the inputs to determine the output [14]. Figure 1 shows an example of an artificial neuron.

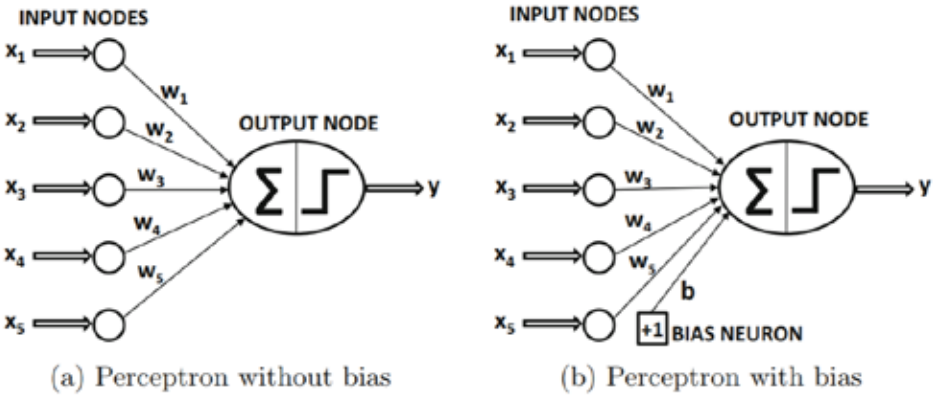


Figure 1. Example of an Artificial Neuron (a) Without Bias and (b) With Bias. Source: [14].

The mathematical equation for the output of a simple artificial neuron can be written as

$$output = \begin{cases} 0 \text{ if } \sum_j w_j x_j \leq threshold \\ 1 \text{ if } \sum_j w_j x_j > threshold \end{cases} . \qquad (2.1.1)$$

The weighted sum of the inputs is compared to a threshold, determining if the output of the artificial neuron is one or zero. If we move the threshold to the other side of the equation, it becomes the bias [15]. Therefore, the bias equals the negative threshold. The new mathematical equation for the artificial neuron can be written as

$$output = \begin{cases} 0 \text{ if } w \cdot x + b \leq 0 \\ 1 \text{ if } w \cdot x + b > 0 \end{cases}. \tag{2.1.2}$$

2. Activation Functions

Nonlinear activation functions are applied to the weighted outputs of the artificial neuron, limiting the value of the outputs. The normal ranges of the outputs from the activation functions are [0,1] or [-1,1]. Nonlinear activation functions allow the NN to learn nonlinear mapping; without them, it is equal to linearly mapping the input and output domains [13]. There are several types of nonlinear activation functions, as shown in Figure 2. Sigmoid, hyperbolic tangent (tanh), and Leaky Rectified Linear Unit (ReLU) activation functions are used in this thesis.

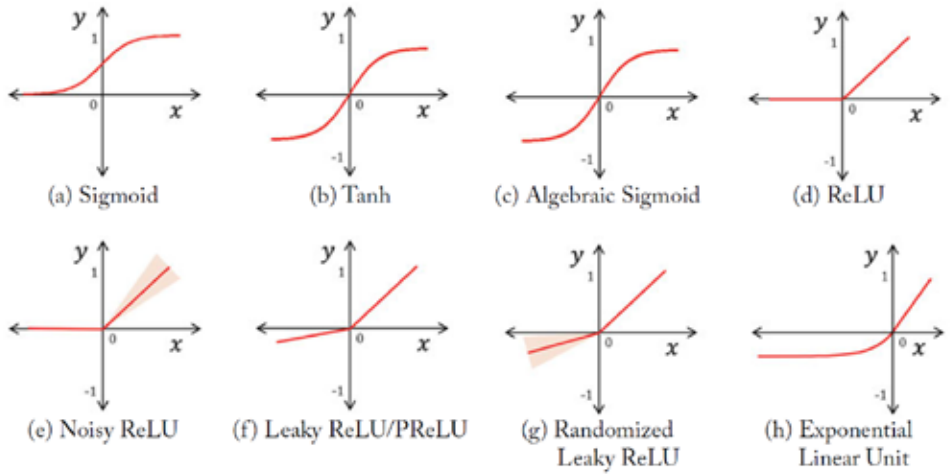


Figure 2. Graphs of Nonlinear Activation Functions. Source: [13].

For classification or categorization, a typical activation function is the softmax activation. The softmax activation converts the real output of the nodes of the last layer

into probabilities. These probabilities determine the predicted category to which the output belongs.

3. Neural Networks

The NN consists of many artificial neurons in layers. The basic NN architecture is an input layer, followed by one or more hidden layers and then an output layer. The output the artificial neurons of the input layer is input into the artificial neurons in the first hidden layer. The output of the artificial neurons of the first hidden layer then feeds the next hidden layer. This process repeats until the output layer containing the loss function is reached [14]. The loss function is used to determine the quality of the output predictions [13]. Depending on the loss function results, the weights of the artificial neurons are adjusted to improve performance. This process is NN training and can require much time, processing power, and a large data set [15].

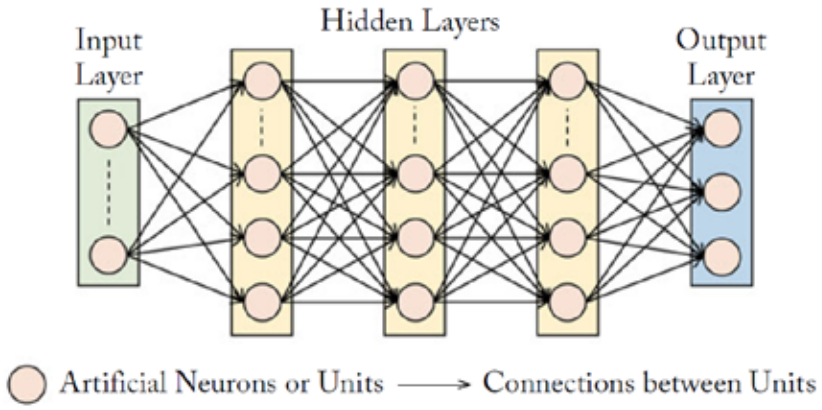


Figure 3. Example Neural Network. Source: [13].

4. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are feed-forward neural networks that only flow in one direction with no loops within the structure [13]. They are used extensively in image classification due to their ability to exploit distinctive features in the images. CNNs differ from other neural networks by their use of convolutional layers in their hidden layers. The convolutional layers consist of filters, also called kernels. These kernels are

convolved with the input to generate the output feature map [13]. The kernel can be imagined as a grid with the size determined by the user. As the grid slides along the data, it is convolved with the input. This process extracts the input data features and gives it a value used as the next layer's input [14]. Figure 4 shows an example of a 7x7 image convolved with a 3x3 kernel.

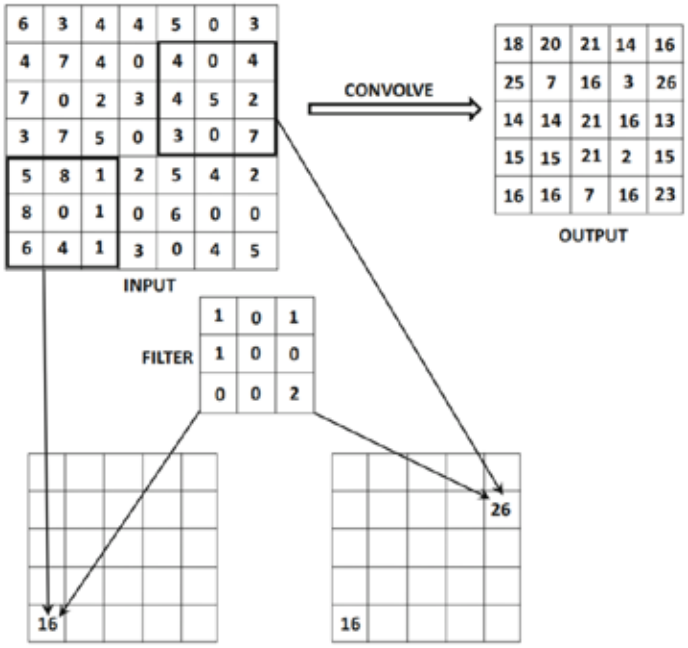


Figure 4. Example of a 7x7 Grid Convolved with a 3x3 Filter. Source: [14].

After the kernel is convolved with the input data, an activation function determines if meaningful features are present. When trained with labeled data, the CNN can learn the meaningful features of the data through this process [13]. This ability makes CNNs valuable networks for classification.

5. Deep Residual Networks

Deep Residual Networks (RNs) are a type of CNN proposed in [2] to classify signals. The RN differs from the typical CNN by using a residual unit instead of a traditional kernel. The unique feature of the residual unit is the skip connection. The skip

connection adds the original input to the output of the convolutional kernel [13]. Figure 5 shows the structure of a residual unit.

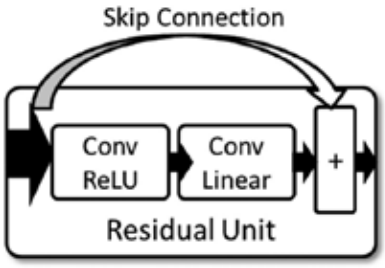


Figure 5. Diagram of a Residual Unit. Adapted from [2].

Using the skip connection improves the ability of the RN to focus on meaningful features in the data. RN networks show promise as a simpler method for CNNs to learn the essential features of the input data [13].

6. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are feedback NNs as opposed to feed-forward NNs. The RNN has short-term memory of the previously processed input. The RNN uses this information to improve its future predictions [14]. The RNN has been useful for work with sequential data, text, biological data, and time-series data where the critical attributes of the data are dependent on its sequence [13]. Shown in Figure 6 is the basic architecture of the RNN.

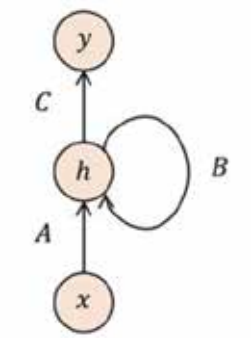


Figure 6. Recurrent Neural Network Architecture. Adapted from [13].

As an RNN executes, it becomes more difficult for the network to remember more than a few previous steps. As an RNN is trained, the essential features present in the data can become lost due to the short memory. Long Short Term Memory Recurrent Neural Networks (LSTM-RNNs) were developed to improve the memory of the RNN [13]. The LSTM-RNN adds an element of long-term memory to the network. The introduction of long-term memory improves the ability of the network to learn the essential features of the data [3].

B. GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks are a recent development in machine learning. Goodfellow first proposed the GAN in 2014 [16]. The GAN consists of two adversarial networks, a generator network and a discriminator network, pitted against each other [16]. This section covers an overview of GANs and will look at the semi-supervised GANs used for classification.

1. Generative Adversarial Networks Overview

The GAN consists of two convolutional neural networks that compete against each other in a minimax competition. The two CNNs are known as the discriminator and the generator. The generator creates fake data from input noise. The goal of the generator is to produce data that will fool the discriminator. The discriminator gives a binary output classifying the data as real or fake [20]. The basic structure of the GAN is shown in Figure 7.

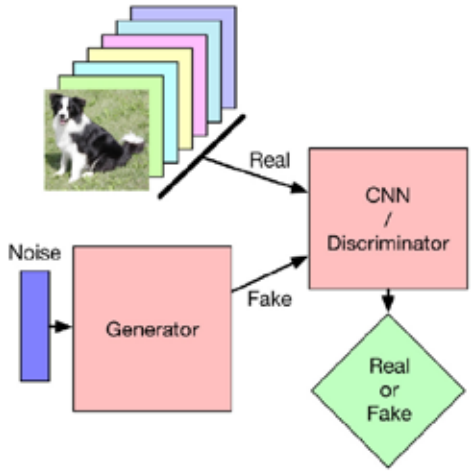


Figure 7. Example of a Generative Adversarial Network. Source: [17].

When first training the GAN, the generator outputs subpar data that does not fool the discriminator because the generator does not know the probability distribution of the real data; the initial data the generator creates does not resemble the real data. When the discriminator correctly identifies the real and fake samples, the generator' parameters are updated in an attempt to improve its performance [19]. As the GAN is trained, the generator begins to learn the correct features through this process. As the generator improves, the discriminator will begin to misclassify data as real or fake. The parameters of the discriminator will then be updated to improve its performance. This game is played until the discriminator accuracy reaches around 50%, and the generator and discriminator are no longer improving [20].

2. Semi-supervised GAN

Different types of GANs are used to solve different types of problem sets. For classification, the SGAN has shown promise. The SGAN requires both unlabeled and a small amount of labeled data to train. Since large amounts of labeled data can be challenging to obtain, this provides a means to train the GAN [17]. The SGAN consists of the typical generator and discriminator, but it also includes a classifier. The classifier can be another NN within the GAN, or, as used in this thesis, it can be stacked with the

discriminator. In a stacked SGAN, the discriminator and the classifier use the same NN, but each has its own output layer [21]. This technique reduces the amount of processing power needed for the GAN.

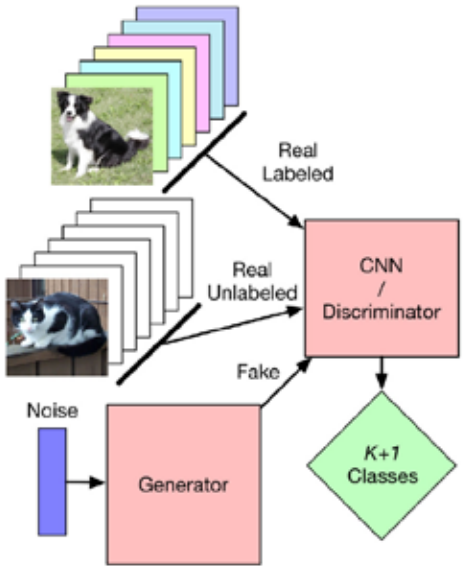


Figure 8. Example of a Semi-Supervised GAN. Source: [17].

Figure 8 shows an example of an SGAN. We can see that the classifier has $K+1$ outputs, where K is the number of classes in the labeled data, and the extra class is for the fake data. Using the unlabeled data prevents the discriminator from overfitting on the labeled data, and the fake data prevents the simple features in the data from becoming the deciding factor in classification [17]. These properties allow the SGAN to be trained to classify data accurately without the need for large amounts of training data.

3. Training the Semi-supervised GAN

When training the NN or GAN, the dataset is split into a training dataset and a testing dataset. The training dataset is used during the training of the GAN. The testing dataset is used to evaluate the performance of the trained GAN [13].

For the training of the SGAN, a portion of the training dataset is labeled. The labeled data is used to train the classifier. The process of training on the labeled data is

referred to as supervised training. The unlabeled portion of the training dataset is used to train the discriminator. The process of training on the unlabeled data is referred to as unsupervised training. Combining these training techniques allows the classifier to learn the features of the classes [21].

Datasets are usually too large to be input into a NN or GAN all at once. Therefore, datasets are separated into batches that are a portion of the total dataset. Each batch is used as an input into the network during training. Training on one batch is called a training iteration. Training on one dataset is called an epoch [22]. The equation for the total training iterations can be written as

$$training\ iterations = \frac{number\ of\ \ snippets\ in\ dataset}{batch\ size} \cdot number\ epochs\ . \quad (2.1.3)$$

THIS PAGE INTENTIONALLY LEFT BLANK

III. RELATED RESEARCH

In this chapter, we look at the use of NNs and GANs in previous research. NNs have been studied for decades and are used in a multitude of ways. We will focus on the use of NNs in the field of signal classification. GANs are a more recent development and have been used primarily to optimize the generator to create realistic fake data. Since the generator is used for training in this thesis, we will focus on research where GANs are used for signal classification.

A. USE OF NEURAL NETWORKS IN SIGNAL CLASSIFICATION

Research into NNs used to classify signals for cognitive radios has been popular in recent years. In 2016, O'Shea et al. [1]. proved a CNN could classify signals by modulation type using only the In-phase and Quadrature (I/Q) values. For their research, O'Shea et al. [20]. created the Radio Machine Learning Dataset (RadioML Dataset) consisting of 11 modulation schemes with SNRs ranging from -20 dB to 18 dB. This data set is used in this thesis and discussed in Chapter IV. O'Shea et al. showed that CNNs could correctly classify signals by modulation schemes across all SNRs with an 87% accuracy, but they showed poor performance at low SNRs.

In 2018, O'Shea et al. used a residual network to classify signal modulations with an extended dataset with 24 modulation schemes [2]. The training data used by O'Shea et al. consisted of 1.44 million snippets of raw I/Q data. After 14 hours of training, the network achieved 95.6% accuracy across all SNRs.

Rajendran et al. classified signals by modulation using an LSTM-RNN as a follow-on to O'Shea et al. research in [3], [1]. Using the RadioML Dataset, the LSTM RNN achieved a 90% classification rate across all SNRs. At low SNRs, below -2 dB, the LSTM RNN used by Rajendran et al. had a 5%-10% improvement in accuracy over the CNN used by O'Shea et al. [1].

Other CNNs have been used to classify signals for transmitter identification through automatic feature learning [4]–[9]. These researchers used their own datasets or existing

datasets instead of the Radio Machine Learning Dataset. Of note, none of these researchers used raw I/Q data in their research.

B. USE OF GANS IN SIGNAL CLASSIFICATION

The use of GANs to classify signals for cognitive radios is relatively new research. In 2018, Tang et al. used an Auxiliary Classifier GAN (AC-GAN) to classify signals by modulation scheme [10]. An SGAN adds a class for fake data; an AC-GAN must first determine if the data is real or fake before classifying it [26]. Tang et al. preprocessed the input signal data into the constellation diagrams of the modulation schemes. The GAN then used image recognition to classify the signals. Tang et al. showed that the AC-GAN improved modulation classification by about 0.1% at high SNRs and around 6% at low SNRs over a traditional CNN.

In 2018, Li et al. [11]. used an AC-GAN to classify signals by modulation scheme. In contrast to Tang et al. [10], Li et al. used the raw I/Q data provided in the RadioML Dataset. Li et al. added an encoder network to improve the efficiency of the GAN and prevent collapse during training. The results showed an improvement of classification accuracy between 0.1% and 12% over a CNN.

In 2019, Roy et al. [12]. used a GAN, not for modulation classification, but for transmitter identification. The goal of Roy et al. was to identify trusted and counterfeit transmitters. Raw I/Q data was used as the input to the GAN. They identified transmitters by using the I/Q imbalance to identify the transmitter. The GAN used by Roy et al. was able to distinguish between the trusted and counterfeit transmitters with an accuracy of 99.9% compared to 81.6% achieved by a CNN.

IV. DATASETS AND SOFTWARE RESOURCES

A. DATASETS

We used two datasets to test the ability of GANs to classify SOIs. We first tested the ability of an SGAN to classify signals by modulation using the RadioML Dataset. A dataset of only Quadrature Phase Shift Keying signals was then created to test the ability of GANs to classify SOIs within signals of the same modulation scheme.

1. Radio Machine Learning Dataset

O'Shea and West created the RadioML dataset to test signal modulation classification [18]. The dataset was created to be used in [1] but has become a standard dataset used for signal classification. O'Shea and West used GNU Radio software to generate signals with 11 different modulation schemes, eight digital and three analog signals. The simulated effects of sample rate offset, center frequency offset, selective fading, and noise were added to the signals. The added noise ranged from -20 dB to 18 dB, increasing in 2 dB increments. Each of the signals was then converted into I/Q components and divided into snippets, each of length 128 discrete-time samples. There were 1,000 snippets created for each modulation at each SNR, totaling a dataset of 220,000 snippets [18].

2. QPSK Datasets

To test the ability of a GAN to classify signals within a single modulation scheme, we created a Quadrature Phase Shift Keying (QPSK) dataset using MATLAB. We created nine signals which began with differing 48-bit addresses. Random bits were added after the addresses to simulate data. Each of the addresses increasingly varied from the address of the first programmed signal. The subsequent signal addresses were varied from the first signal address by a 12.5%, 25%, 33.3%, 50%, 66.7%, 75%, and 100% bit differences.

The signals were then modulated, filtered using a root-raised cosine filter, and used to modulate a carrier signal. Noise was then added to the resulting RF signals with SNRs ranging from -20 dB to 18 dB, increasing in 2 dB increments. The signals were then

downconverted into I/Q components. The sampling rate was ten samples per symbol. Different length snippets of each of the signals were taken and used to create datasets. Three datasets were created, one with snippets each 128 discrete-time samples long, one with snippets each 256 discrete-time samples long, and the last with snippets each 512 discrete-time samples long. There were 1,000 snippets created for each signal at each SNR, resulting in three datasets, each containing 180,000 snippets.

B. SOFTWARE RESOURCES

For this thesis, primarily free, open-source software was used. For the GAN, TensorFlow with Keras was used to code the NNs within it. Google Colaboratory was used as the environment to run the networks as it provides the increased processing power needed to handle the large amount of data used.

1. TensorFlow and Keras

TensorFlow is free, open-source machine learning software developed by Google and released in 2015. TensorFlow provides a flexible architecture that can run across multiple CPUs and GPUs. In TensorFlow, dataflow graphs represent the computations, shared states, and operations that mutate the state. TensorFlow can be used with various applications with a focus on deep learning and NNs [23].

We used Keras to code the GANs for signal classification. Keras is an application program interface that runs on top of TensorFlow. It provides deep learning and NN libraries that enable easy coding of networks in Python. Keras allows the user to construct NNs by stacking sequential layers or models. Each layer or model is one operation of the NN. Keras allows a complicated NN or GAN to be easily constructed without the need for extensive coding [24].

2. Google Colaboratory

Google Colaboratory, ‘Colab’ for short, was used to create and run the GANs for this thesis. Colab is a hosted Jupyter notebook that provides the ability to write and execute Python code from a browser. Colab is especially useful for machine learning as it provides computing resources, including GPUs, that improve the speed of training a GAN. Colab

also easily connects with Google Drive for the storage of data and code. The combination of Colab and Google Drive provided a simple and easy method to upload datasets, download results, and have the computational power needed to run the GAN with the larger datasets [25].

THIS PAGE INTENTIONALLY LEFT BLANK

V. SGAN ARCHITECTURE AND EXPERIMENT METHODOLOGY

A. ARCHITECTURE OF THE SGANS

For this research, the GANs used for each experiment were kept as similar as possible to provide a consistent way to evaluate the performance against the datasets. All the GANs used a stacked discriminator network consisting of a discriminator and a classifier. The same generator configuration was used for all the GANs.

1. Stacked Discriminator

The discriminator of each GAN was stacked, so the discriminator and classifier used the same CNN with different output layers. The discriminators consisted of three convolutional layers with Leaky ReLU activation functions. Each convolutional layer used a 1x3 sized kernel filter. After the convolutional layers, the data was flattened and input into a dense layer with an output equal to the number of modulation classes or signals used in the dataset.

As seen in Figure 9, the output of the dense layer was the input to the discriminator activation function and the classifier activation function. The discriminator activation function used the sigmoid activation function and determined if the image is real or fake. The output of the dense layer was also the input into the classifier using a softmax activation function. The output of the classifier activation function was a positive integer whose maximum value equals the number of classes or signals being classified.

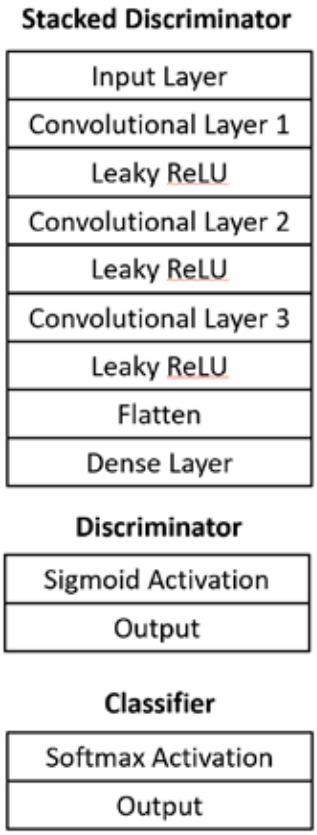


Figure 9. Stacked Discriminator Layers

2. Generators

The generators used in the GANs were all of the same architecture. The number of input nodes and the size of the generators output were the only variables between the GANs. The input nodes were first connected to a dense layer with a Leaky ReLU activation function. Two transposed convolutional layers followed the dense layer. Each of these transposed convolutional layers used a Leaky ReLU activation function. The final layer was a convolutional layer with a hyperbolic tangent activation function.

Each of the transposed convolutional layers and the regular convolutional layer used 3x3 sized kernels. After the final layer, the output data was reshaped to match the snippet size used in the datasets. The layers are the generator are shown in Figure 10.

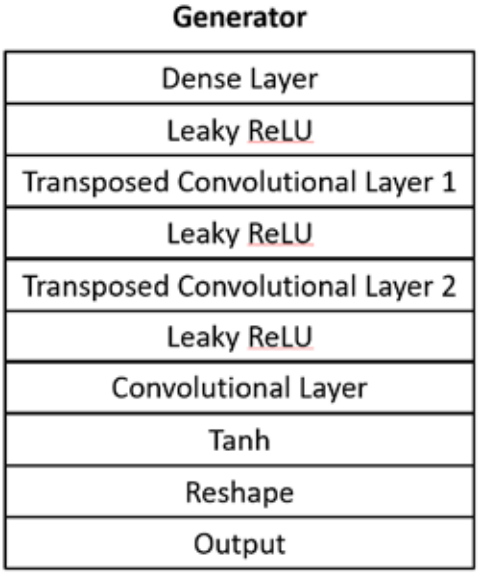


Figure 10. Generator Layers

3. Data Handling

For each of the GANs, the data was handled the same. The input data from the datasets were first normalized, so their values were between [-1,1]. This normalization helps stabilize the NN and allows the generator output to resemble the input data closely [14]. For each dataset, 25%, 50%, and 75% of the dataset were chosen to be the training data, leaving the rest for testing. Using the RadioML dataset, 25% is 55,000 snippets, 50% is 110,000 snippets, and 75% is 165,000 snippets. Using the QPSK dataset, 25% is 45,000 snippets, 50% is 90,000 snippets, and 75% is 135,000 snippets. Out of the training data, 1,000 snippets were selected as the labeled data for supervised training.

4. GAN Comparisons

The GANs required some parameter changes due to the size of each input signal and the different categories for classification in the datasets. To simplify the naming convention, the GANs were labeled SGAN 1 through SGAN 4. SGAN 1 used the RadioML dataset. It had input snippets each of length of 128 discrete-time samples and an output of 11 different modulation schemes. SGAN 2 through SGAN 4 used the QPSK data set, which

contains signals with nine different addresses. SGAN 2, SGAN 3, and SGAN 4 use input snippets each of length 128, 256, and 512 discrete-time samples, respectively.

Table 1 and Table 2 show each layer of the SGAN stacked discriminator inputs and outputs. They also show the number of parameters in each layer. As shown in the tables, the number of parameters significantly increases as the snippets' length increases. This results in a much longer training time and computing power needed for the longer snippets.

Table 1. Comparison of Discriminator Layers of the SGANs

	SGAN 1		SGAN 2		SGAN 3		SGAN 4	
Layer (type)	Output (size)	Parameters	Output (size)	Parameters	Output (size)	Parameters	Output (size)	Parameters
Input	2, 128	0	2, 128	0	2, 256	0	2, 512	0
Reshape	2, 128, 1	0	2, 128, 1	0	2, 256, 1	0	2, 512, 1	0
Convolutional Layer 1	2, 128, 256	1024	2, 128, 256	1024	2, 256, 512	2048	2, 512, 1024	4096
LeakyReLU	2, 128, 256	0	2, 128, 256	0	2, 256, 512	0	2, 512, 1024	0
Convolutional Layer 2	2, 128, 128	98342	2, 128, 128	98342	2, 256, 256	39472	2, 512, 512	1573376
LeakyReLU	2, 128, 128	0	2, 128, 128	0	2, 256, 256	0	2, 512, 512	0
Convolutional Layer 3	2, 128, 64	24640	2, 128, 64	24640	2, 256, 128	98432	2, 512, 128	196736
LeakyReLU	2, 128,64	0	2, 128,64	0	2, 256, 128	0	2, 512, 128	0
Flatten	16384	0	16384	0	65536	0	131072	0
Dense	11	180235	9	145161	9	589833	9	1179657
Custom Activation	1	0	1	0	1	0	1	0
Total Parameters		304331		269257		1083785		2953865

Table 2. Comparison of Classifier Layers of the SGANs

	SGAN 1		SGAN 2		SGAN 3		SGAN 4	
Layer (type)	Output (size)	Parameters	Output (size)	Parameters	Output (size)	Parameters	Output (size)	Parameters
Input	2, 128	0	2, 128	0	2, 256	0	2, 512	0
Reshape	2, 128, 1	0	2, 128, 1	0	2, 256, 1	0	2, 512, 1	0
Convolutional Layer 1	2, 128, 256	1024	2, 128, 256	1024	2, 256, 512	2048	2, 512, 1024	4096
LeakyReLU	2, 128, 256	0	2, 128, 256	0	2, 256, 512	0	2, 512, 1024	0
Convolutional Layer 2	2, 128, 128	98342	2, 128, 128	98342	2, 256, 256	39472	2, 512, 512	1573376
LeakyReLU	2, 128, 128	0	2, 128, 128	0	2, 256, 256	0	2, 512, 512	0
Convolutional Layer 3	2, 128, 64	24640	2, 128, 64	24640	2, 256, 128	98432	2, 512, 128	196736
LeakyReLU	2, 128,64	0	2, 128,64	0	2, 256, 128	0	2, 512, 128	0
Flatten	16384	0	16384	0	65536	0	131072	0
Dense	11	180235	9	145161	9	589833	9	1179657
Softmax	11	0	9	0	9	0	9	0
Total Parameters		304331		269257		1083785		2953865

The generator number of parameters also increases significantly as the snippet length increases. As shown in Table 3, as the length of the snippet doubles, the total number of parameters quadruples.

Table 3. Comparison of Generators Layers of the SGANs

	SGAN 1		SGAN 2		SGAN 3		SGAN 4	
Layer (type)	Output (size)	Parameters	Output (size)	Parameters	Output (size)	Parameters	Output (size)	Parameters
Input	100	0	100	0	100	0	100	0
Dense	8192	827392	8192	827392	32768	3309568	131072	13238272
LeakyReLU	8192	0	8192	0	32768	0	131072	0
Reshape	1, 64, 128	0	1, 64, 128	0	1, 128, 256	0	1, 256, 512	0
Transpose Conv 1	1, 64, 128	147584	1, 64, 128	147584	1, 128, 256	590080	1, 256, 512	2359808
LeakyReLU	1, 64, 128	0	1, 64, 128	0	1, 128, 256	0	1, 256, 512	0
Transpose Conv 2	2, 128, 128	147584	2, 128, 128	147584	2, 256, 256	590080	2, 512, 512	2359808
LeakyReLU	2, 128, 128	0	2, 128, 128	0	2, 256, 256	0	2, 512, 512	0
Convolutional Layer	2, 128, 1	1153	2, 128, 1	1153	2, 256, 1	2305	2, 512, 1	4609
Reshape	2, 128	0	2, 128	0	2, 256	0	2, 512	0
Total Parameters		123713		123713		4492033		17962497

B. EXPERIMENT METHODOLOGY

In this section, we look at the setup of each experiment to test the performance of the SGANs in classification. Classification accuracy and training time will be the primary performance measures in determining the effectiveness of the SGANs. The dataset, training dataset percentage, and the snippet length of the data will be varied.

1. Experiment 1

In Experiment 1, we test the ability of the SGAN to classify signals by modulation type. SGAN 1 was set up to accept the snippets each of length 128 discrete-time samples provided in the RadioML dataset. We ran SGAN 1 using training data consisting of 25%, 50%, and 75% of the total dataset. At the end of each epoch, the classifier, which had been trained only on labeled data, was tested on all the training data for classification accuracy. We then used the classifier model with the best accuracy to evaluate the performance of SGAN 1. We used the overall training time, overall classification accuracy across all SNRs, and the classification accuracy at each SNR to judge the effectiveness of SGAN 1 to classify modulation type at each dataset training percentage.

2. Experiment 2

For Experiment 2, we tested the ability of SGAN 2 to classify signals within a modulation type. We used the QPSK dataset with snippets each of length 128 discrete-time samples in this experiment. SGAN 2 was tested using 25%, 50%, and 75% of the overall dataset as a training dataset. As in Experiment 1, we tested the classifier at the end of each

epoch to check its accuracy. The classifier models were then saved, and after training, the model with the highest accuracy was used to evaluate the performance of SGAN 2 on the testing data. We used the training time, overall classification accuracy across all SNRs, and classification accuracy at each SNR to judge the effectiveness of the SGAN 2 to classify the signals at each dataset training percentage.

3. Experiment 3

In Experiment 3, we used the QPSK dataset with the snippets each of length 256 discrete-time samples to evaluate SGAN 3. The experiment was set up the same as Experiment 2, using 25%, 50%, and 75% of the overall dataset for training. We tested and saved the classifier at the end of each epoch as in the previous experiments. The effectiveness of SGAN 3 at classification was evaluated on the performance measures as in the previous experiments. The expected outcome of increasing the snippet sample length was increased accuracy at the cost of increased training time.

4. Experiment 4

For Experiment 4, we performed the same experiments as in Experiment 2 and Experiment 3, but we used the QPSK data with snippets each of length 512 discrete-time samples. For the training of SGAN 4, we used 25%, 50%, and 75% of the overall data as a training dataset as before. As in the other experiments, we tested and saved the classifier after each epoch. We also used the same performance measures of training time, overall classification accuracy across all SNRs, and classification accuracy at each SNR. The increase in snippet sample length should increase accuracy and training time.

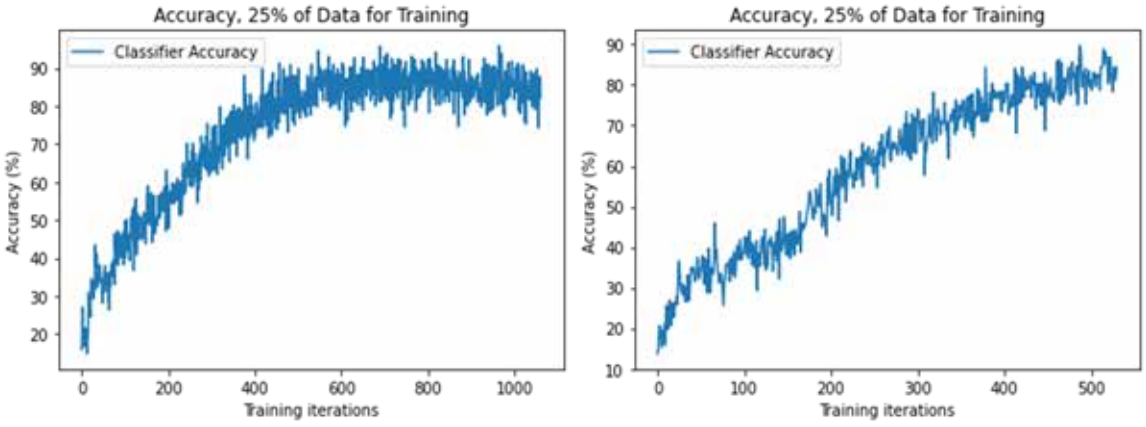
VI. RESULTS AND ANALYSIS

In this chapter, we discuss the results of the experiments. We present the data collected from each experiment. Using the performance measures discussed in Chapter V, we analyze the results to determine the effectiveness of the SGAN in modulation and signal classification.

A. RESULTS

1. Experiment 1

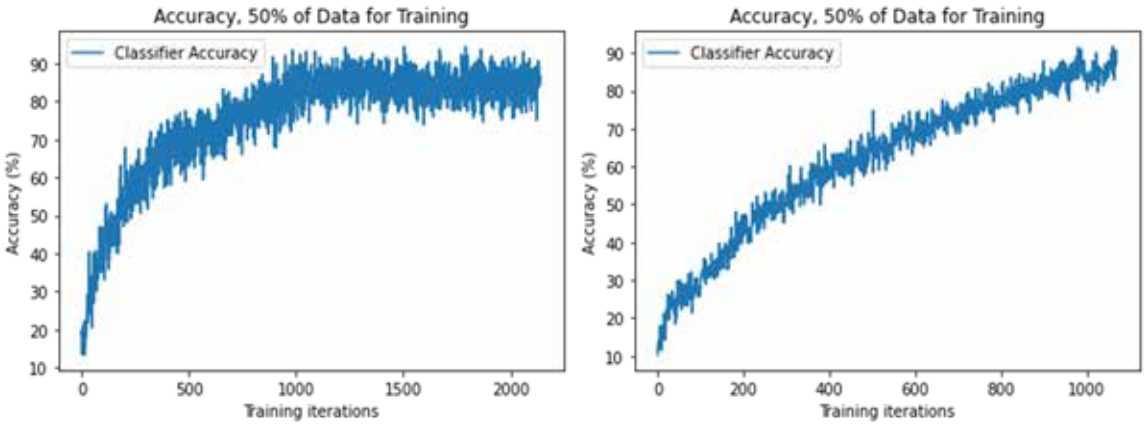
For Experiment 1, we divided the RadioML dataset into training and testing datasets. SGAN 1 was first trained using 25% of the RadioML data as the training dataset. This portion contained random snippets of each signal at each SNR value between -20 dB and 18 dB. We first chose 20 epochs to train the class with a batch size of 1024 snippets. At 25%, this resulted in 1060 training iterations. From Figure 11, we can see that during training, the classifier improvement after the first ten epochs is negligible. Therefore, we reduced the number of epochs to ten, 530 iterations, and trained SGAN 1 again. Training for ten epochs, the classifier achieves around 85% accuracy on the labeled data.



The left plot shows SGAN 1 classifier accuracy on labeled data when trained for 20 epochs on 25% of the dataset. The right plot shows SGAN 1 classifier accuracy on labeled data when trained for ten epochs on 25% of the dataset.

Figure 11. SGAN 1 Classifier Accuracy on Labeled Data Using 25% of the Dataset for Training

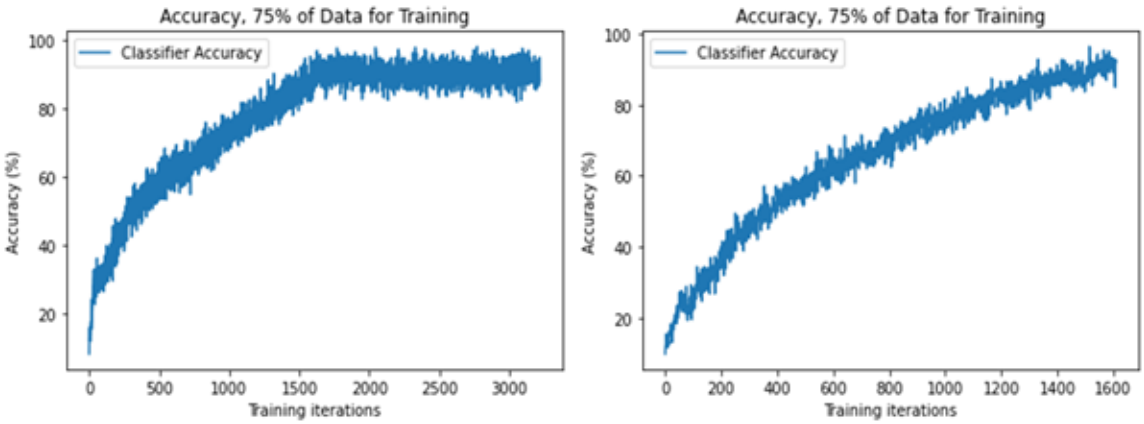
SGAN 1 was then trained using 50% of the dataset for training. We first trained using 20 epochs and a batch size of 1024, equaling 2140 training iterations. Figure 12 shows the classifier accuracy remains consistent at around ten epochs. We reduced the number of epochs to ten, resulting in 1070 training iterations, and trained SGAN 1 again. The training accuracy reached around 86% accurate classification on the labeled data.



The left plot shows SGAN 1 classifier accuracy on labeled data when trained for 20 epochs on 50% of the dataset. The right plot shows SGAN 1 classifier accuracy on labeled data when trained for ten epochs on 50% of the dataset.

Figure 12. SGAN 1 Classifier Accuracy on Labeled Data Using 50% of the Dataset for Training

Finally, we trained SGAN 1 on 75% of the dataset. Using a batch size of 1024 and 20 epochs resulted in 3220 training iterations. Figure 13 shows that after around ten epochs, the accuracy improvement is negligible. The training was reduced to 10 epochs with 1610 iterations total. Using ten epochs, the classifier accuracy is around 95% on labeled data.



The left plot shows SGAN 1 classifier accuracy on labeled data when trained for 20 epochs on 75% of the dataset. The right plot shows SGAN 1 classifier accuracy on labeled data when trained for ten epochs on 75% of the dataset.

Figure 13. SGAN 1 Classifier Accuracy on Labeled Data Using 75% of the Dataset for Training

After each epoch, the classifier was tested on the training dataset, and the classifier model was saved. The best performing classifier from each training was then used to classify the training and testing datasets. Table 4 shows the results of the classification of the training and testing datasets.

Table 4. SGAN 1 Accuracy on Training and Testing Datasets

Overall Accuracy of Classifier			
Percent of Data Used for Training	25%	50%	75%
Training Dataset	80.12	82.16	81.90
Testing Dataset	80.50	82.09	82.24

Each classifier was then run on the data to test its performance at different SNRs. Figure 14 shows the accuracy of the classifiers trained. As seen in Figure 14, there appears to be little variation between the performances of the classifiers trained on the different amounts of data. Using 75% of the dataset only slightly improves accuracy at low SNRs but shows no consistent improvement at high SNRs. The data displayed in this plot and similar plots in this chapter are included in the appendix.

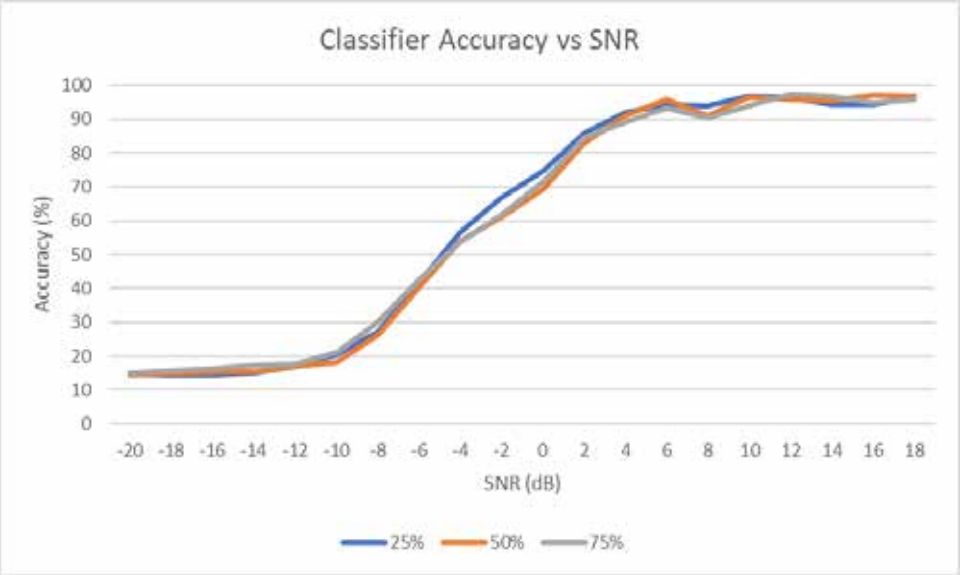


Figure 14. SGAN 1 Classifier Accuracy at Different SNRs

Finally, the time required to train SGAN 1 is shown in Table 5. As the amount of data used for training increased, the training time and computational power needed to train also increased.

Table 5. SGAN 1 Training Times

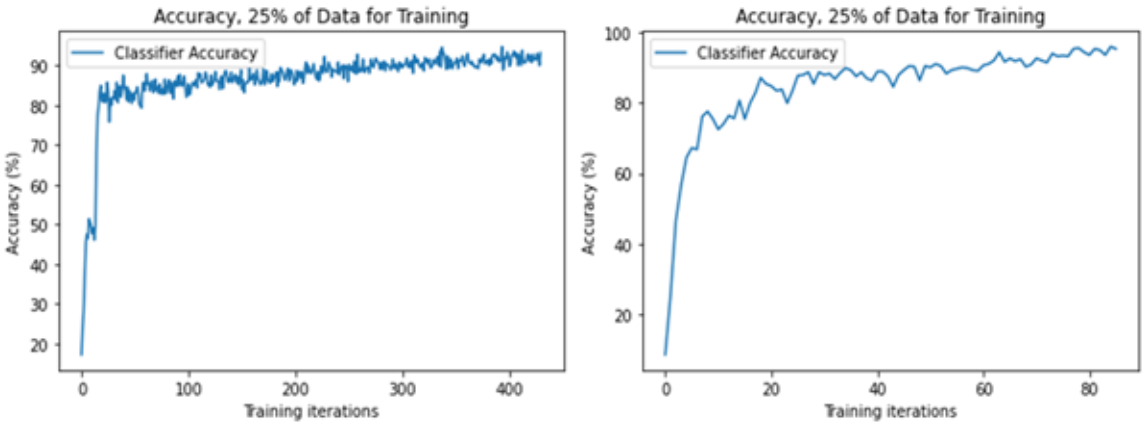
Training Time

Percentage of Dataset	Time (min)
25%	7
50%	14
75%	24

2. Experiment 2

For Experiment 2, we used the QPSK dataset with snippets each of length 128 discrete-time samples to evaluate SGAN 2. SGAN 2 was first trained using 25% of the dataset for training. Using the knowledge from Experiment 1, we trained SGAN 2 for ten epochs with a batch size of 1024. At 25%, this resulted in 430 training iterations. From Figure 15, we can see that during training, the classifier accuracy quickly increased and

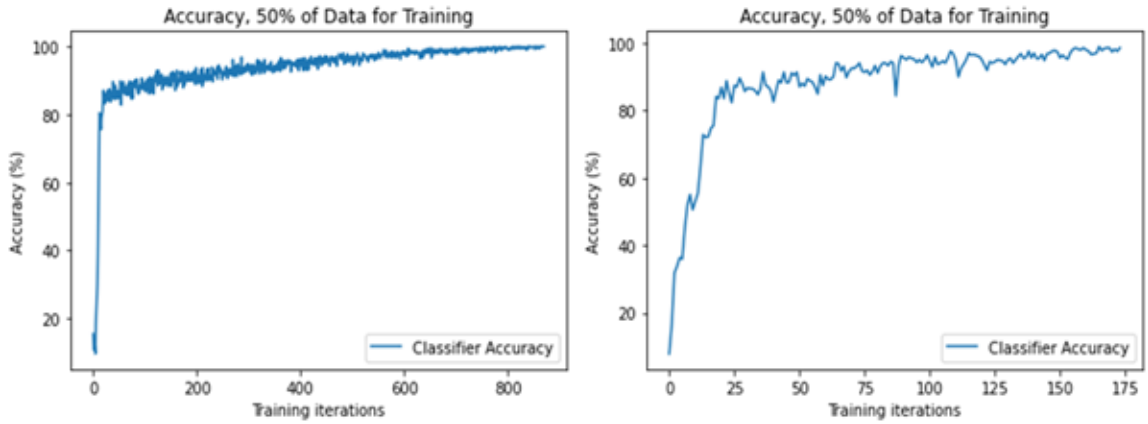
stabilized at approximately 90% accuracy. We then reduced the number of training epochs to two, 86 training iterations, and the classifier accuracy was consistent at approximately 90% on the labeled data.



The left plot shows SGAN 2 classifier accuracy on labeled data when trained for ten epochs on 25% of the dataset. The right plot shows SGAN 2 classifier accuracy on labeled data when trained for two epochs on 25% of the dataset.

Figure 15. SGAN 2 Classifier Accuracy on Labeled Data Using 25% of the Dataset for Training

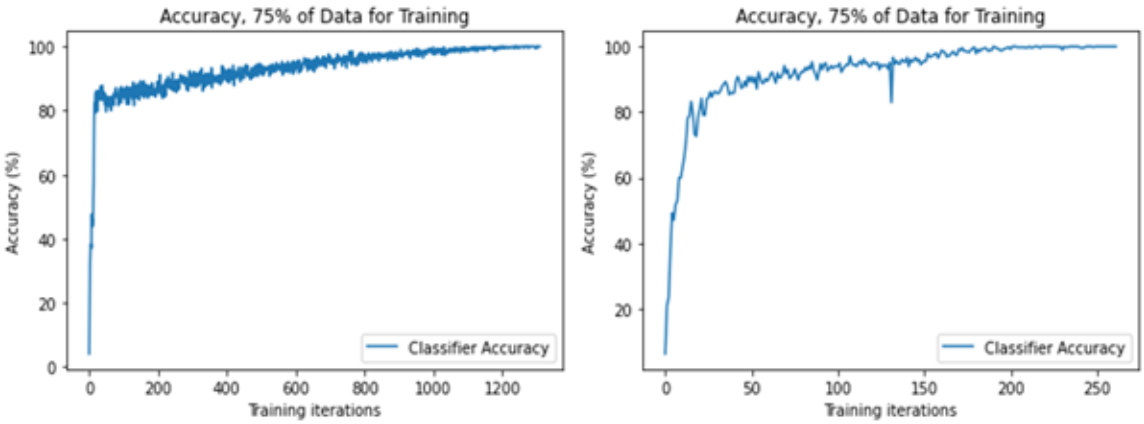
The percentage of training data used was then increase to 50%. Using batches of size 1024 and ten epochs resulted in 870 training iterations. As shown in Figure 16, there is minimal improvement in the classifier accuracy on the labeled data after around two epochs. Therefore, we reduced the number of epochs to two, resulting in 174 training iterations. The classifier accuracy quickly rises to around 90% and then slowly increases to nearly 100% on the labeled data.



The left plot shows SGAN 2 classifier accuracy on labeled data when trained for ten epochs on 50% of the dataset. The right plot shows SGAN 2 classifier accuracy on labeled data when trained for two epochs on 50% of the dataset.

Figure 16. SGAN 2 Classifier Accuracy on Labeled Data Using 50% of the Dataset for Training

SGAN 2 was then run using 75% of the dataset to train. Once again, a batch size of 1024 and 10 epochs was used for the initial training. This resulted in 1310 training iterations. As shown in Figure 17, the accuracy increases rapidly early and then slowly increases to around 100%. We then reduced the number of epochs to two with 262 training iterations. The classifier accuracy showed the same performance with two epochs as with ten, also shown in Figure 17.



The left plot shows SGAN 2 classifier accuracy on labeled data when trained for ten epochs on 75% of the dataset. The right plot shows SGAN 2 classifier accuracy on labeled data when trained for two epochs on 75% of the dataset.

Figure 17. SGAN 2 Classifier Accuracy on Labeled Data Using 75% of the Dataset for Training

After each epoch, the classifier was tested on the training dataset, and the classifier model was saved. The best performing classifier from each training was then used to classify the training and testing datasets. The classifiers trained on two epochs were selected because of their advantage in the required training time. The accuracy of the classifiers is shown in Table 6. From Table 6, we can see that using 50% and 75% of the dataset performs slightly better than using 25%, but there is very little difference between 50% and 75%.

Table 6. SGAN 2 Accuracy on Training and Testing Datasets

Overall Accuracy of Classifier

Percent of Data Used for Training	25%	50%	75%
Training Dataset	83.44	84.36	84.70
Testing Dataset	82.56	85.20	84.24

The SGAN 2 classifiers were then run on the testing dataset to test their accuracy at different SNR levels. Figure 18 shows that using 50% and 75% of the dataset for training results in a slightly better performance at low SNRs. All three of the classifiers reach 99% classification accuracy or better by 0 dB SNR.

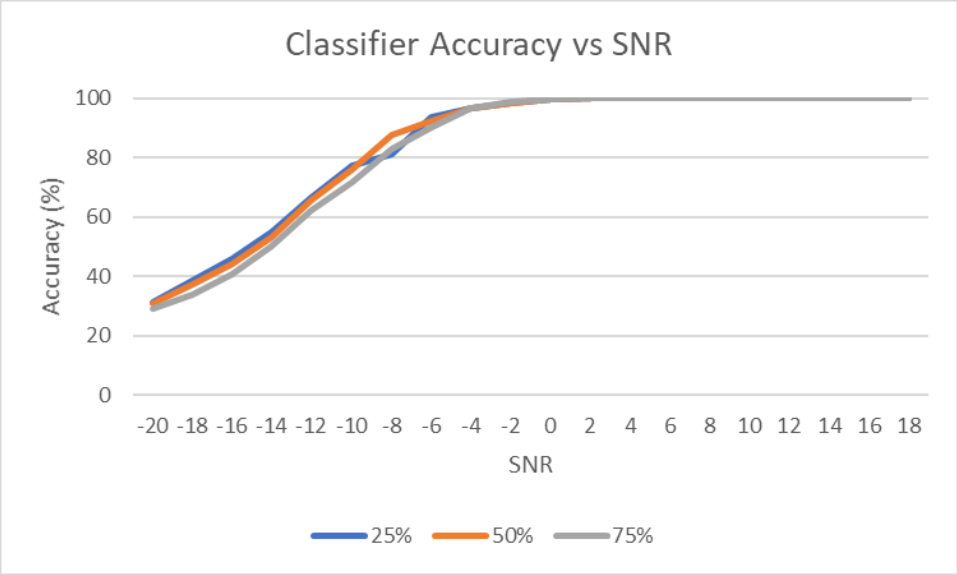


Figure 18. SGAN 2 Classifier Accuracy at Different SNRs

The last metric evaluated in Experiment 2 was the training times. Table 7 shows the variation in training times using 25%, 50%, and 75% of the dataset and two epochs for training. As the amount of data used increases, the training time increases. Since there is little difference in the classification capabilities of using 50% and 75% of the dataset for training, the training time would indicate that using 50% of the dataset would be more practical.

Table 7. SGAN 2 Training Times

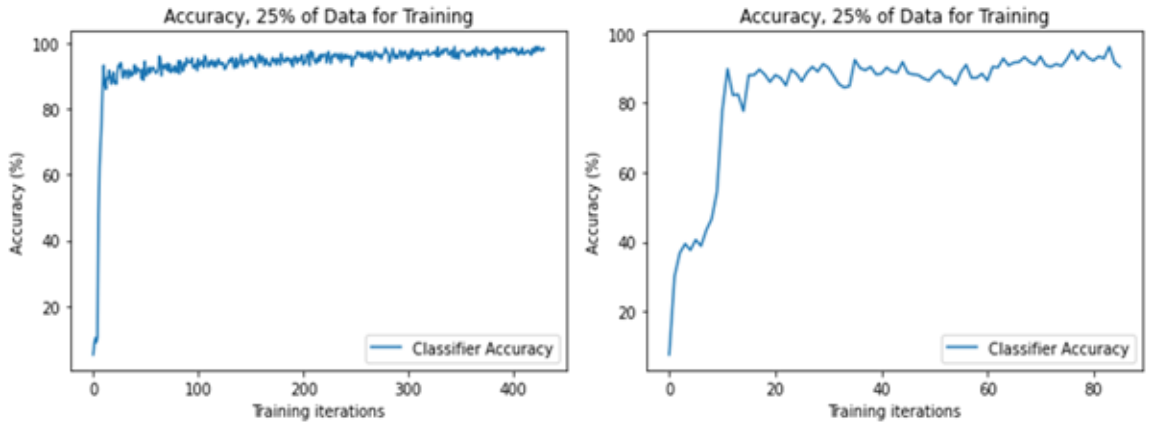
Training Time

Percentage of Dataset	Time (min)
25%	3.5
50%	6.5
75%	12

3. Experiment 3

For Experiment 3, the QPSK dataset with snippets each of length 256 discrete-time samples was split into training and testing sets. We first trained SGAN 3 using 25% of the dataset for training. We initially trained SGAN 3 for ten epochs with a batch size of 1024

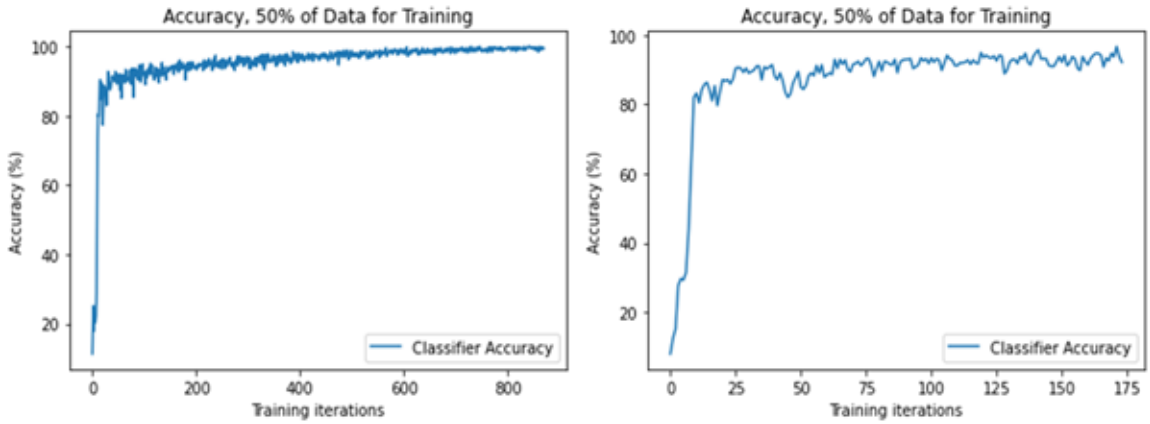
snippets. At 25%, this resulted in 430 training iterations. From Figure 19, we can see that during training, the classifier accuracy quickly rises to about 90% and continues to increase slowly to around 100% accuracy. We then reduced the number of epochs to two with 86 training iterations. There was a negligible difference in the classifier performance at two epochs and ten epochs, as shown in Figure 19.



The left plot shows SGAN 3 classifier accuracy on labeled data when trained for ten epochs on 25% of the dataset. The right plot shows SGAN 3 classifier accuracy on labeled data when trained for two epochs on 25% of the dataset.

Figure 19. SGAN 3 Classifier Accuracy on Labeled Data Using 25% of the Dataset for Training

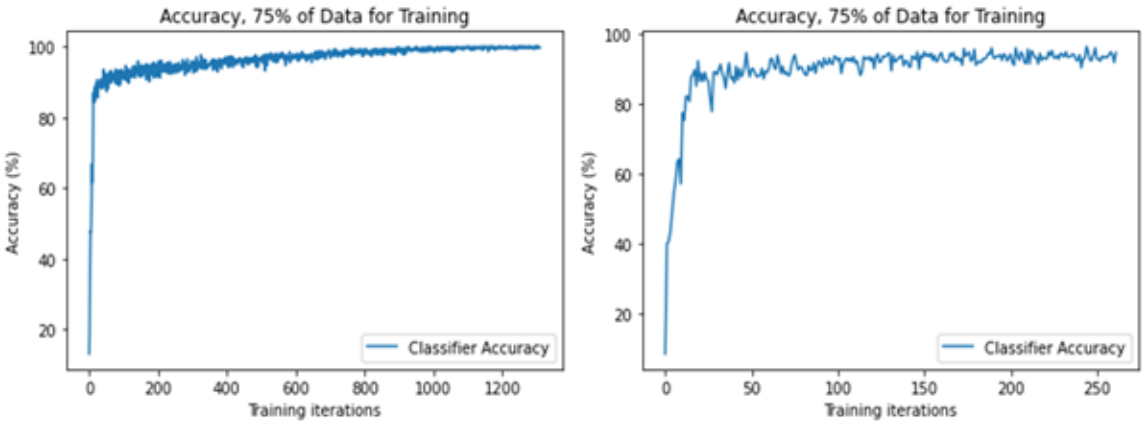
SGAN 3 was then trained using 50% of the dataset for training. Using a batch size of 1024 and ten epochs resulted in a total of 870 training iterations. The classifier accuracy rose quickly to around 95% accuracy. We then reduced the number of epochs to two for 174 training iterations. Figure 20 shows, the classifier accuracy on the labeled training data with two epochs was near 95%, the same as with ten epochs.



The left plot shows SGAN 3 classifier accuracy on labeled data when trained for ten epochs on 50% of the dataset. The right plot shows SGAN 3 classifier accuracy on labeled data when trained for two epochs on 50% of the dataset.

Figure 20. SGAN 3 Classifier Accuracy on Labeled Data Using 50% of the Dataset for Training

We then ran SGAN 3 using 75% of the dataset to train. A batch size of 1024 and 10 epochs was used during training resulting in 1310 training iterations. As shown in Figure 21, the accuracy increases rapidly early and then slowly increases to around 98% accuracy on the labeled training data. We then ran SGAN 3 using only two epochs for training; i.e., 262 training iterations. The classifier performance on the labeled training data was on par with the performance observed using ten epochs.



The left plot shows SGAN 3 classifier accuracy on labeled data when trained for ten epochs on 75% of the dataset. The right plot shows SGAN 3 classifier accuracy on labeled data when trained for two epochs on 75% of the dataset.

Figure 21. SGAN 3 Classifier Accuracy on Labeled Data Using 75% of the Dataset for Training

As before, after each epoch, the classifier was tested on the training dataset, and the classifier model was saved. With the negligible difference in performance between using ten and two epochs for training, we chose two epochs because of the lower training time. The best performing classifier from each training run was then used to classify the training and testing datasets. The accuracy of the classifiers is shown in Table 8. From Table 8, we can see that when using snippets of length 256 discrete-time samples, there appears to be a negligible difference in the performance of the classifiers using 25%, 50%, and 75% of the dataset for training.

Table 8. SGAN 3 Accuracy on Training and Testing Datasets

Overall Accuracy of Classifier

Percent of Data Used for Training	25%	50%	75%
Training Dataset	89.99	90.16	89.40
Testing Dataset	89.79	89.81	89.20

Each classifier was then run on the testing dataset to assess its performance at different SNRs. Figure 22 shows the performance of the trained classifiers. As shown in Figure 22, there appears to be little variation between the performances of the classifiers

when trained on the different data amounts. All three classifiers reach 99% accuracy or above by -4 dB SNR.

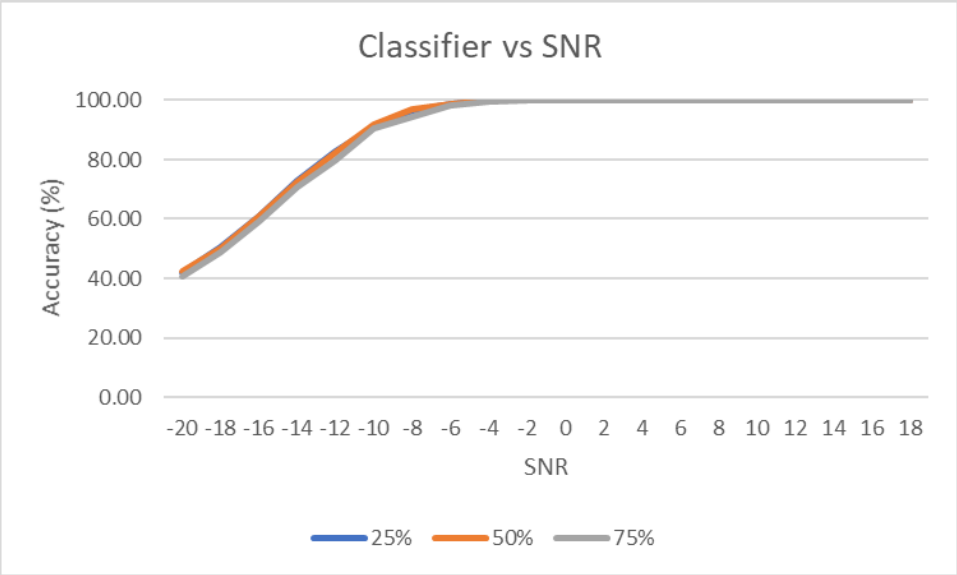


Figure 22. SGAN 3 Classifier Accuracy at Different SNRs

The training times for SGAN 3 are recorded in Table 9. Table 9 shows the variation in training times using 25%, 50%, and 75% of the dataset for training. As the amount of data used increased, the training time significantly increased. Since there was only a slight difference in the classification capabilities of using 25%, 50%, or 75% of the dataset for training, the training time would indicate that using 25% of the dataset would be most practical.

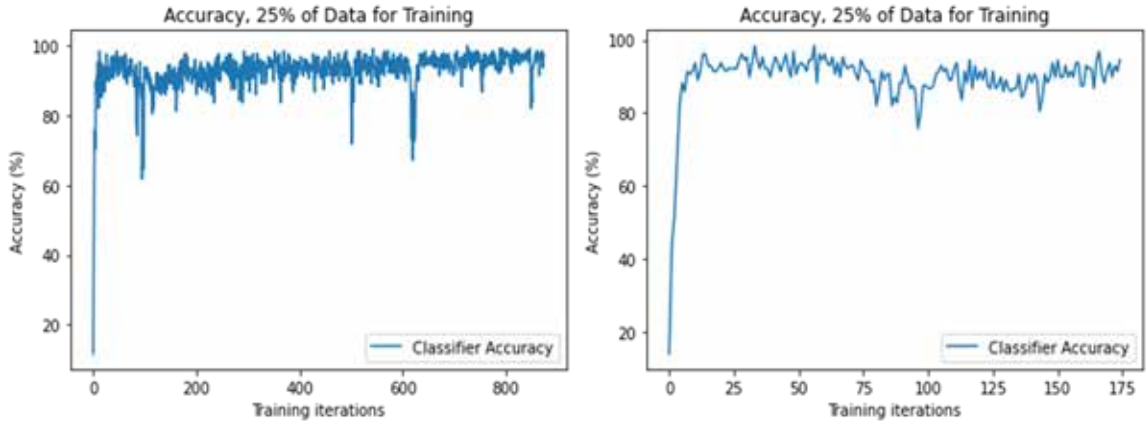
Table 9. SGAN 3 Training Times

Training Time

Percentage of Dataset	Time (min)
25%	4.5
50%	9
75%	16

4. Experiment 4

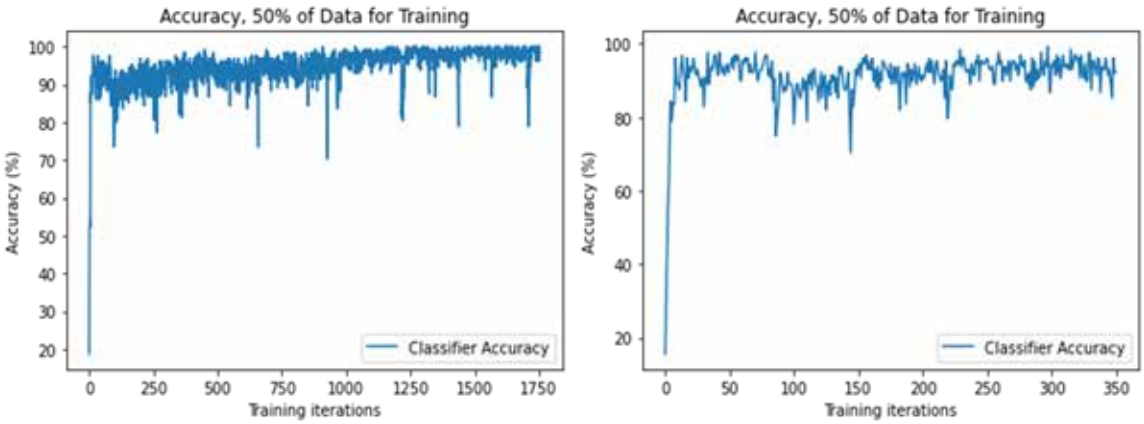
For Experiment 4, the QPSK dataset with snippets each of length 512 discrete-time samples was split into training and testing sets. SGAN 4 was first trained using 25% of the dataset for training. Due to the software limitations of Google Colab, we had to reduce the batch size and the number of epochs from 1024 snippets and ten epochs to 256 snippets and five epochs. At 25%, this resulted in 875 training iterations. From Figure 23, we can see that during training, the classifier accuracy immediately rises to above 90% on the labeled data. We then trained SGAN 4 with one training epoch, resulting in 175 training iterations. There was a negligible difference in the performance of the classifier trained in one epoch and the one trained in five epochs.



The left plot shows SGAN 4 classifier accuracy on labeled data when trained for five epochs on 25% of the dataset. The right plot shows SGAN 4 classifier accuracy on labeled data when trained for one epoch on 25% of the dataset.

Figure 23. SGAN 4 Classifier Accuracy on Labeled Data Using 25% of the Dataset for Training

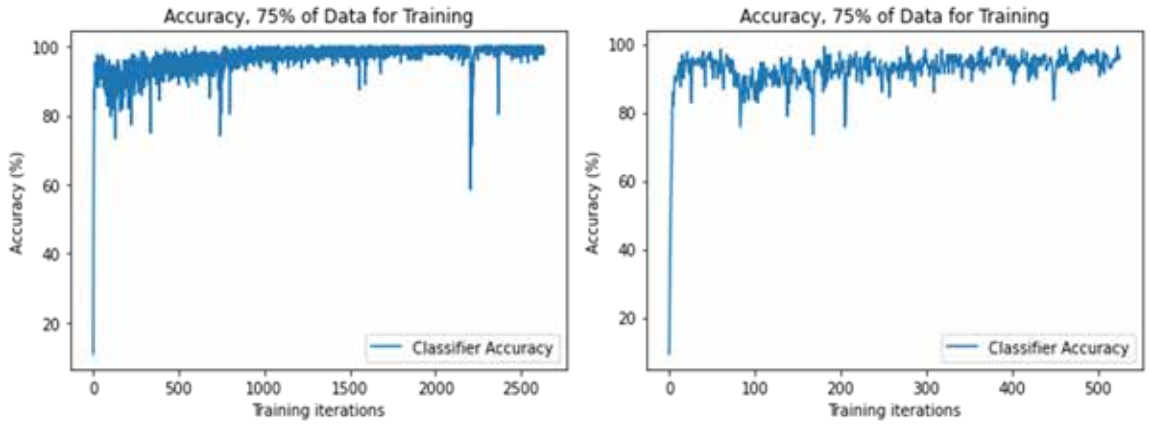
SGAN 4 was then trained using 50% of the dataset for training. Using a batch size of 256 and five epochs resulted in a total of 1755 training iterations. Figure 24 shows the accuracy during training follows the same pattern as 25% of the dataset and increases quickly to around 95% on the labeled data. We then trained SGAN 4 for one epoch, resulting in 351 training iterations. As shown in Figure 24, training SGAN 4 for one epoch yielded results similar to training it for five epochs.



The left plot shows SGAN 4 classifier accuracy on labeled data when trained for five epochs on 50% of the dataset. The right plot shows SGAN 4 classifier accuracy on labeled data when trained for one epoch on 50% of the dataset.

Figure 24. SGAN 4 Classifier Accuracy on Labeled Data Using 50% of the Dataset for Training

We then ran SGAN 4 using 75% of the dataset for training. A batch size of 256 and five epochs was used during training resulting in 2635 training iterations. As shown in Figure 25, the accuracy increases rapidly early and then slowly increases to around 100% on the labeled data. We then trained SGAN 4 using one epoch; i.e., 527 training iterations. When training with one epoch, the classifier results were similar to those when using five epochs.



The left plot shows SGAN 4 classifier accuracy on labeled data when trained for five epochs on 75% of the dataset. The right plot shows SGAN 4 classifier accuracy on labeled data when trained for one epoch on 75% of the dataset.

Figure 25. SGAN 4 Classifier Accuracy on Labeled Data Using 75% of the Dataset for Training

After each epoch, the classifier was tested on the training dataset, and the classifier model was saved. We chose to use the classifiers trained in one epoch due to the significant training time advantage. The best performing classifiers were then evaluated on the training and testing dataset and not just the labeled data. The accuracy of the classifiers is shown in Table 10. From Table 10, we can see the overall accuracy of the classifiers was not significantly affected by the training dataset size used.

Table 10. SGAN 4 Accuracy on Training and Testing Datasets

Overall Accuracy of Classifier

Percent of Data Used for Training	25%	50%	75%
Training Dataset	88.75	89.71	89.65
Testing Dataset	88.39	89.18	89.32

The classifiers from SGAN 4 were then run on the testing data to assess their accuracy at different SNR levels. Figure 26 shows that using 50% and 75% of the training results in a slightly better performance below -8 dB. All three of the classifiers reach 99% classification accuracy or better by -4 dB SNR.

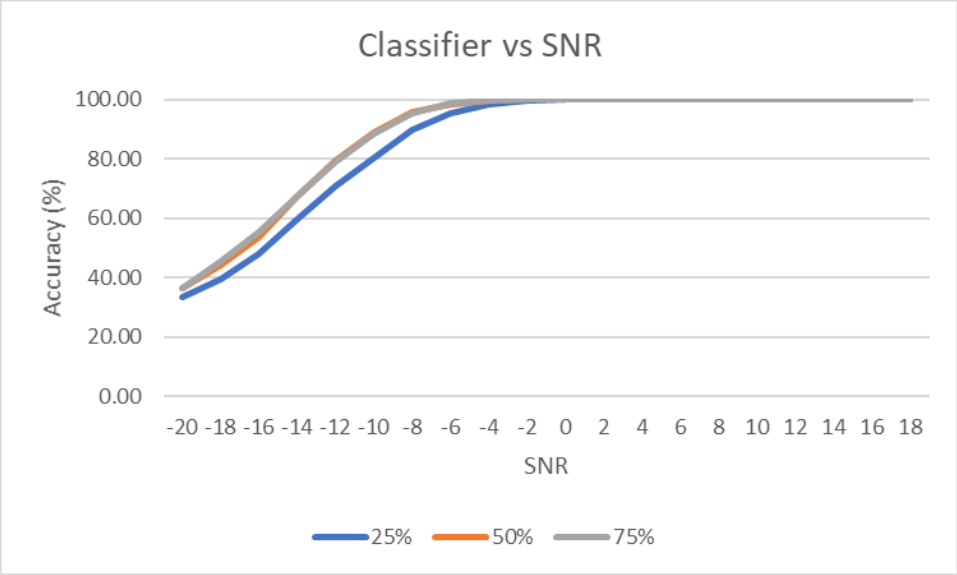


Figure 26. SGAN 4 Classifier Accuracy at Different SNRs

Table 11 shows the recorded training times for SGAN 4. The increase in snippet length to 512 discrete-time samples significantly increased the training time required. Since there was only a negligible difference in the classification capabilities of using 25%, 50%, or 75% of the dataset for training, the training time would indicate that using 25% of the dataset for training would be most practical when using snippets of length 512 discrete-time samples.

Table 11. SGAN 4 Training Times

Training Time

Percentage of Dataset	Time (min)
25%	14
50%	24
75%	47

B. ANALYSIS

This section covers the ability of the SGAN to classify signals by modulation and the ability of the SGAN to classify distinct QPSK signals. The basis for this analysis is the data from the experiments in the previous section.

1. Classification of Signals by Modulation Type

The SGAN proved to have the ability to classify signals by modulation type. As shown in Table 4, the SGAN 1 was able to classify the signals across all SNRs between 80.12% and 82.24%. As will be discussed in a following section SGAN 1 performed well at high SNRs but poorly at low SNRs.

a. Effect of Training Dataset Size and SNR on Classification

The training dataset size did not significantly affect the performance of the classifier from SGAN 1. As shown in Table 4, there was only a 2% difference overall. SNR had a significant effect on the classifier accuracy. Figure 14 shows that below -4 dB SNR, the classifier accuracy was below 50% regardless of the amount of training data used. Also shown in Figure 14, the classifier accuracy reaches around 90% at 4 dB SNR. From the results, we conclude the SNR of the signals is the most significant factor affecting the ability of an SGAN classifier to classify signals by modulation scheme accurately.

b. Effect of Training Dataset Size on Training Time

The training dataset size used by SGAN 1 to train the classifier did not affect its performance, but it significantly impacted the time needed to train. As shown in Table 5, as the training dataset size increases from 25% to 75% of the dataset, training time increases from seven to 24 minutes. For a 2% increase in performance, there was a 342% increase in training time. The time cost of training for a small increase in performance may not be worth it, depending on the application.

2. Classification of QPSK Signals

Experiments 2, 3, and 4 evaluated the ability of an SGAN-trained classifier to classify signals within the same modulation scheme. For the experiments, we used the

QPSK signal dataset discussed earlier. The SGAN trained classifiers were able to classify the QPSK signals accurately.

a. Effect of Training Data Size on Classification

The training dataset size used in the training of the SGAN classifier had little impact on the overall performance of the classifier. The average overall classification accuracy performed by SGAN 2, SGAN 3, and SGAN 4 on the training and testing datasets are shown in Table 12. The average accuracy is between 87.39% and 88.08% on the training dataset and 86.91 and 88.06% on the testing dataset. These results indicate the percentage of the dataset used for training did not affect classifier performance. The results showed that in order to affect the performance significantly, the training dataset would need to be even smaller than 25%.

Table 12. Average Classifier Accuracy by Training Dataset Size

	Average Accuracy of Classifier		
Percent of Data Used for Training	25%	50%	75%
Training Dataset	87.39	88.08	87.92
Testing Dataset	86.91	88.06	87.59

b. Effect of Snippet Length on Classification

As shown in Table 13, the SGAN 3 and SGAN 4 classifiers, which used snippets of length 256 and length 512 discrete-time samples, respectfully, performed equally well with accuracies around 89%. SGAN 2 used the snippets of length 128 discrete-time samples and achieved 84.17% accuracy on the training dataset and 84.00% accuracy on the testing dataset. These results indicate that an SGAN classifier trained using snippets of length 256 or length 512 discrete-time samples should be used for training.

Table 13. Average Classifier Accuracy by Snippet Length

Average Accuracy of Classifier		
	Training Data	Testing Data
SGAN 2 (128)	84.17	84.00
SGAN 3 (256)	89.85	89.60
SGAN 4 (512)	89.37	88.96

c. Effect of Training Dataset Size and Snippet Length on Classification at Different SNRs

From Table 14, we can determine varying the training dataset size does not improve performance of the SGAN classifier at the various SNR levels. Table 14 also shows that the snippet length does affect the SGAN classifier accuracy at low SNRs. SGAN 3, using snippets of length 256 discrete-time samples, performed the best at low SNRs, followed SGAN 4, using snippets of length 512 discrete-time samples. Around 0 dB SNR, all of the classifiers start to perform essentially the same. These results showed that using snippets of length 256 discrete-time samples would be the optimal selection to train the classifier when compared across all SNRs.

Table 14. Classification Accuracy by SNR, Training Dataset Size, and Snippet Length

Classification Accuracy

SNR vs. Training Dataset Size

SNR	25%	50%	75%
-20	35.56	36.49	35.33
-18	42.91	44.03	42.87
-16	51.57	52.94	51.68
-14	62.63	64.36	62.76
-12	73.46	75.86	73.69
-10	82.99	85.64	83.61
-8	89.26	93.44	91.02
-6	95.90	96.57	95.76
-4	98.25	98.81	98.67
-2	99.44	99.53	99.56
0	99.85	99.91	99.91
2	99.99	99.99	100.00
4	100.00	100.00	100.00
6	100.00	100.00	100.00
8	100.00	100.00	100.00
10	100.00	100.00	100.00
12	100.00	100.00	100.00
14	100.00	100.00	100.00
16	100.00	100.00	100.00
18	100.00	100.00	100.00

SNR vs. Snippet Length

SNR	SGAN 2	SGAN 3	SGAN 4
-20	30.42	41.67	35.28
-18	36.79	49.95	43.08
-16	43.64	60.23	52.32
-14	52.85	72.10	64.80
-12	64.88	81.77	76.36
-10	75.12	91.14	85.98
-8	83.98	96.04	93.70
-6	92.13	98.59	97.51
-4	96.84	99.67	99.22
-2	98.72	99.92	99.88
0	99.69	99.99	99.99
2	99.98	100.00	100.00
4	100.00	100.00	100.00
6	100.00	100.00	100.00
8	100.00	100.00	100.00
10	100.00	100.00	100.00
12	100.00	100.00	100.00
14	100.00	100.00	100.00
16	100.00	100.00	100.00
18	100.00	100.00	100.00

25%, 50%, and 75% represent the amount of the dataset used for training.

SGAN 2 represents the snippet length of 128 samples.

SGAN 3 represents the snippet length of 256 samples.

SGAN 4 represents the snippet length of 512 samples.

d. Effect of Training Dataset Size and Data Snippet Length on Training Times

The most significant impact of the training dataset size and the snippet length was on the training times. As the training dataset size moves from 25% to 50%, the time increase on average was 155% and a 483% increase in training time from 50% to 75%, as shown in Table 15.

Table 15. Average Increase in Training Time vs. Dataset Size Increase

Increase in Dataset Size	Increase in Time (%)
25% - 50%	155
50% - 75%	483

The average increase in training time due to increased snippet length was 134% when increasing from snippets of length 128 discrete-time samples to snippets of length 256 discrete-time samples. The average increase in training time when comparing the snippets of length 256 discrete-time samples to snippets of length 512 discrete-time samples was 288%, as shown in Table 16. The results indicate that using the smallest size training data set and shortest snippet length requires the least training time. Combined with the performance accuracy associated with different snippet lengths, the optimal SGAN would use snippets of length 256 discrete-time samples and use 25% of the dataset for training.

Table 16. Average Increase in Training Time vs. Snippet Size Increase

Increase in Snippet Length	Increase in Time (%)
128 - 256	134
256 - 512	288

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND FUTURE WORK

In this chapter, we discuss answers to original research questions posed in Chapter I. The results and analysis provided in Chapter VI are the basis for the conclusions. After the conclusions, future research is suggested.

A. CONCLUSIONS

1. Classifying Signals by Modulation Type

The first question posed was the ability of the SGAN to classify signals by modulation type using only I/Q data. Experiment 1 with the SGAN 1 proved that the SGAN could perform this task with an accuracy of approximately 80%. These percentages were an overall average disregarding training dataset size and signal SNRs.

This research showed that there was very little difference between using 25% of the dataset or 75% of the data to train the SGAN. Overall, there is only a 2% performance difference between them. The amount of training data needed for efficient and accurate classification is 25% of the dataset or below.

The SGAN was very effective in classifying signals by modulation when the SNR was above 4 dB but was poor at classifying signals when the SNR was -4 dB or below. At 4 dB SNR and above, the SGAN was able to classify at an accuracy of over 90%. At -4 dB, SNR and below, the SGAN classified signals at an accuracy of 50% or below.

The length of time used to train the SGAN for classification by modulation varied with the training dataset size. The shortest training time was 7 min when using 25% of the dataset for training and the longest training time was 24 minutes when using 75% of the dataset for training.

The results suggest that out of the three variants of the SGAN used for signal modulation classification, the optimal option would be to use 25% of the dataset for training. This option had the shortest training time with essentially the same performance as the SGANs using 50% and 75% of the dataset for training.

2. Classifying Signals within a Single Modulation Type

The SGAN was able to classify signals within a dataset consisting of a single modulation type using only I/Q inputs. A QPSK dataset consisting of signals with nine different 48-bit addresses was created. The SGAN performed exceptionally well at classifying these signals.

The research showed minimal difference in the performance of the SGAN when using 25%, 50%, or 75% of the dataset for training. The best performance of the SGAN classifier, regardless of the snippet length or SNR, occurred using 50% of the data for training. Although this was the best performer, there was only a 1% difference in accuracy between all three datasets.

The most significant factor in SGAN classifier performance was the length of the snippets. Using the snippets of length 256 discrete-time samples outperformed using either the snippets of length 128 or 512 discrete-time samples. The difference between 128 and 256 was 5% accuracy, and between 256 and 512 was only 0.5%.

Regardless of the training dataset size and snippet length, the SGANs performed almost identically at each SNR. All of the SGAN classifiers were at or above 99% accurate at 0 dB SNR and above. This performance was exceptionally high. A more varied single modulation dataset, incorporating variations expected at a typical receiver, could reduce the accuracy.

Training time for the single modulation SGANs varied greatly as the dataset size and length of the snippets increased. SGAN 2, using the 128 discrete-time sample length, varied between 3.5 minutes at 25% of the dataset and 7 minutes at 75% of the dataset. SGAN 3, using the 256 discrete-time sample length, varied between 4.5 minutes at 25% of the dataset and 13 minutes at 75% of the dataset. SGAN 4, using the 512 discrete-time sample length, varied between 14 minutes at 25% of the dataset and 42 minutes at 75% of the dataset.

Overall, the SGAN was able to classify signals within a single modulation scheme for the task required in this research question. The best performance was demonstrated with SGAN 3, using the snippets of 256 length discrete-time samples and 25% of the data set

for training. This configuration variant had a short training time without sacrificing accuracy.

B. FUTURE WORK

We have shown in this research that the SGAN has the ability not only to classify signal by modulation but to identify an SOI within a modulation type. The GAN architecture shows promise as a way to train an NN to identify an SOI. Improvement to a GAN could be made using a different architecture such as an AC-GAN or improving the existing SGAN by adding more layers. The training of the SGAN could be made more efficient by adding an early stop feature when the accuracy has stopped improving.

The single modulation dataset could be expanded to make the classification more challenging. The dataset used only white Gaussian Noise as an impairment. More impairments such as sample rate offset, center frequency offset, and selective fading could be added to the dataset. The initial phase of the signal could be arbitrary, and jitter could be introduced into the start time. Over-the-air signals could be collected to create a more realistic dataset. Another dataset using a more complicated modulation scheme could also be created to test the limits of the SGAN ability to classify signals.

In addition to making the dataset more dynamic, the amount of training data required could be explored. At our smallest size, we used 25% of the dataset to train the SGAN. The performance of the classifiers indicates that the amount of dataset used to train could be much lower.

Finally, a system could be created to separate and store signals of interest for further processing after they are identified and classified. This feature would add a level of utility to the classifying of the signals of interest.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: CLASSIFIER ACCURACY AT SNRS

The classifier accuracy for each SGAN at each SNR between -20 dB and 18 dB are plotted in Figure 14, Figure 18, Figure 22, and Figure 26. These plots show the graphical representation of the data below.

Table 17 shows the SGAN 1 classifier accuracy at each SNR given the percentage of the dataset used for training.

Table 17. SGAN 1 Classifier Accuracy at Different SNRs

Classifier Accuracy at Each SNR

SNR	25%	50%	75%
-20	14.67	14.34	15.04
-18	14.41	15.12	15.74
-16	14.48	15.29	16.52
-14	15.16	15.41	17.25
-12	17.14	17.02	17.80
-10	20.52	18.41	21.01
-8	27.17	26.22	30.03
-6	41.61	40.44	42.55
-4	56.56	53.88	53.73
-2	66.70	61.16	61.52
0	74.84	69.38	71.62
2	85.82	83.19	84.58
4	92.02	90.96	89.00
6	94.05	95.85	93.30
8	93.89	90.88	90.40
10	96.63	96.59	94.00
12	96.47	95.85	97.47
14	94.35	95.55	96.61
16	94.19	96.93	94.88
18	96.72	96.68	95.65

Table 18 shows the SGAN 2 classifier accuracy at each SNR given the percentage of the dataset used for training.

Table 18. SGAN 2 Classifier Accuracy at Different SNRs

Classifier Accuracy at Each SNR

SNR	25%	50%	75%
-20	29.44	32.77	30.51
-18	34.20	39.23	38.84
-16	40.96	45.42	47.40
-14	50.41	56.77	54.11
-12	61.74	68.67	68.85
-10	71.70	78.61	77.69
-8	82.37	89.86	88.71
-6	90.66	94.42	94.57
-4	95.98	98.01	97.57
-2	98.40	99.23	98.97
0	99.66	99.98	99.78
2	99.11	100.00	99.96
4	100.00	100.00	100.00
6	100.00	100.00	100.00
8	100.00	100.00	100.00
10	100.00	100.00	100.00
12	100.00	100.00	100.00
14	100.00	100.00	100.00
16	100.00	100.00	100.00
18	100.00	100.00	100.00

Table 19 shows the SGAN 3 classifier accuracy at each SNR given the percentage of the dataset used for training.

Table 19. SGAN 3 Classifiers Accuracy at Different SNRs

Classifier Accuracy at Each SNR			
SNR	25%	50%	75%
-20	39.93	39.07	40.30
-18	48.96	46.85	47.37
-16	58.22	56.92	58.35
-14	70.86	69.40	69.52
-12	81.73	79.42	81.77
-10	90.07	88.75	90.07
-8	96.22	94.91	95.87
-6	98.60	97.64	98.24
-4	99.69	99.66	99.47
-2	99.97	99.91	99.12
0	100.00	99.98	99.96
2	100.00	100.00	100.00
4	100.00	100.00	100.00
6	100.00	100.00	100.00
8	100.00	100.00	100.00
10	100.00	100.00	100.00
12	100.00	100.00	100.00
14	100.00	100.00	100.00
16	100.00	100.00	100.00
18	100.00	100.00	100.00

Table 20 shows the SGAN 4 classifier accuracy at each SNR given the percentage of the dataset used for training.

Table 20. SGAN 4 Classifiers Accuracy at Different SNRs

Classifier Accuracy at Each SNR			
SNR	25%	50%	75%
-20	34.14	34.67	35.39
-18	41.84	42.73	41.92
-16	51.50	51.67	50.88
-14	63.32	64.54	63.32
-12	74.88	73.90	64.04
-10	84.67	86.14	75.12
-8	92.86	94.55	86.57
-6	96.89	98.00	97.89
-4	99.09	99.58	99.60
-2	99.93	100.00	100.00
0	99.99	100.00	100.00
2	100.00	100.00	100.00
4	100.00	100.00	100.00
6	100.00	100.00	100.00
8	100.00	100.00	100.00
10	100.00	100.00	100.00
12	100.00	100.00	100.00
14	100.00	100.00	100.00
16	100.00	100.00	100.00
18	100.00	100.00	100.00

LIST OF REFERENCES

[1] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," *arXiv:1602.04105*, Jun. 2016. [Online]. Available: http://arxiv.org/abs/1602.04105.

[2] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, Feb. 2018, doi: 10.1109/JSTSP.2018.2797022.

[3] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, Sep. 2018, doi: 10.1109/TCCN.2018.2835460.

[4] C. Morin, L. Cardoso, J. Hoydis, J.-M. Gorce, and T. Vial, "Transmitter classification with supervised deep learning," *arXiv:1905.07923 [cs, eess]*, May 2019. [Online]. Available: http://arxiv.org/abs/1905.07923.

[5] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "ORACLE: Optimized radio clAssification through convolutional neuraL nEtworks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Apr. 2019, pp. 370–378, doi: 10.1109/INFOCOM.2019.8737463.

[6] K. Youssef, L. Bouchard, K. Haigh, J. Silovsky, B. Thapa, and C. V. Valk, "Machine learning approach to RF transmitter identification," *IEEE Journal of Radio Frequency Identification*, vol. 2, no. 4, pp. 197–205, Dec. 2018, doi: 10.1109/JRFID.2018.2880457.

[7] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for RF device fingerprinting in cognitive communication networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 160–167, Feb. 2018, doi: 10.1109/JSTSP.2018.2796446.

[8] S. Chen, S. Zheng, L. Yang, and X. Yang, "Deep learning for large-scale real-world ACARS and ADS-B radio signal classification," *IEEE Access*, vol. 7, pp. 89256–89264, 2019, doi: 10.1109/ACCESS.2019.2925569.

[9] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, Sep. 2018, doi: 10.1109/MCOM.2018.1800153.

[10] B. Tang, Y. Tu, Z. Zhang, and Y. Lin, “Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks,” *IEEE Access*, vol. 6, pp. 15713–15722, 2018, doi: 10.1109/ACCESS.2018.2815741.

[11] M. Li, O. Li, G. Liu, and C. Zhang, “Generative adversarial networks-based semi-supervised automatic modulation recognition for cognitive radio networks,” *Sensors*, vol. 18, no. 11, Art. no. 11, Nov. 2018, doi: 10.3390/s18113913

[12] D. Roy, T. Mukherjee, M. Chatterjee, E. Blasch, and E. Pasiliao, “RFAL: Adversarial learning for RF transmitter identification and classification,” *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2019, doi: 10.1109/TCCN.2019.2948919.

[13] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, “A guide to convolutional neural networks for computer vision,” *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, Feb. 2018, doi: 10.2200/S00822ED1V01Y201712COV015.

[14] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, 1st ed. 2018. Cham, Switzerland: Springer, 2018.

[15] M. A. Nielsen, *Neural Networks and Deep Learning*. 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[16] I. J. Goodfellow *et al.*, “Generative adversarial networks,” *arXiv:1406.2661 [cs, stat]*, Jun. 2014, Accessed: May 17, 2020. [Online]. Available: http://arxiv.org/abs/1406.2661.

[17] G. Olmschenk, Z. Zhu, and H. Tang, “Generalizing semi-supervised generative adversarial networks to regression using feature contrasting,” *Computer Vision and Image Understanding*, vol. 186, pp. 1–12, Sep. 2019, doi: 10.1016/j.cviu.2019.06.004.

[18] T. J. O’Shea and N. West, “Radio machine learning dataset generation with GNU radio,” *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, Art. no. 1, Sep. 2016. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/11.

[19] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and Tensorflow 2, Third edition*. Birmingham, England: Packt, 2019.

[20] J. Brownlee, "A gentle introduction to generative adversarial networks (GANs)," *Machine Learning Mastery*, June 17, 2019. [Online]. Available: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/.

[21] J. Brownlee, " How to implement a semi-supervised GAN (SGAN) from scratch in Keras," *Machine Learning Mastery*, July 24, 2019. [Online]. Available: https://machinelearningmastery.com/semi-supervised-generative-adversarial-network/.

[22] J. Brownlee, " Difference between a batch and an epoch in a neural network," *Machine Learning Mastery*, October 26, 2019. [Online]. Available: Difference Between a Batch and an Epoch in a Neural Network (machinelearningmastery.com)

[23] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," *arXiv:1603.04467 [cs]*, Mar. 2016, Accessed: May 21, 2020. [Online]. Available: http://arxiv.org/abs/1603.04467.

[24] K. Team, "Keras documentation: About Keras." Accessed: February 6, 2021. [Online]. Available: https://keras.io/about/.

[25] "Colaboratory – Google." Accessed February 6, 2021. [Online]. Available: https://research.google.com/colaboratory/faq.htm.

[26] J. Brownlee, " How to develop an auxiliary classifier GAN (AC-GAN) from scratch with Keras," *Machine Learning Mastery*, Jul. 18, 2019. [Online]. Available: https://machinelearningmastery.com/how-to-develop-an-auxiliary-classifier-gan-ac- gan-from-scratch-with-keras/.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California